รายงานฉบับสมบูรณ์

โครงงานระบบฐานข้อมูล

เรื่อง Campground Booking

จัดทำโดย

กลุ่มที่ 8 “Sigma”

6733167521	พสิษฐ์	บุญโสภณ
6733185821	เพ็ญพิชชา	ปิยาวรานนท์
6733255021	ศิริกาญจน์	ฟักศรีเมือง
6733284221	อติภัทร	บูรณวัฒนาโชค
6733288821	อภิวิชญ์	แสงเพชร
6733293921	อิศศยาพรรณ	ลิ้มม่วงนิล

รายงานนี้เป็นส่วนหนึ่งของรายวิชา 2110322 ระบบฐานข้อมูล

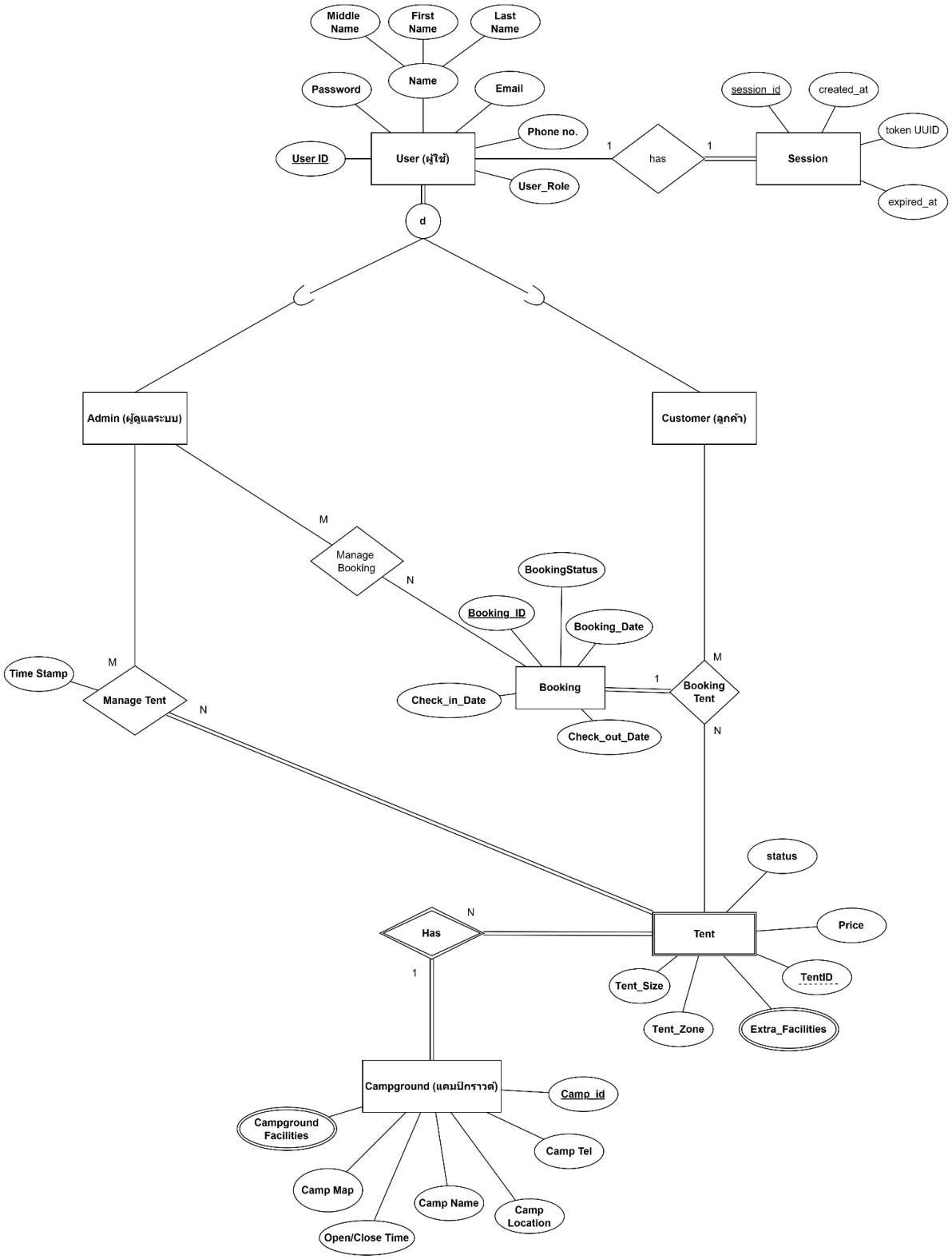
ภาคเรียนที่ 2 ปีการศึกษา 2567

คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์และเทคโนโลยีดิจิทัล

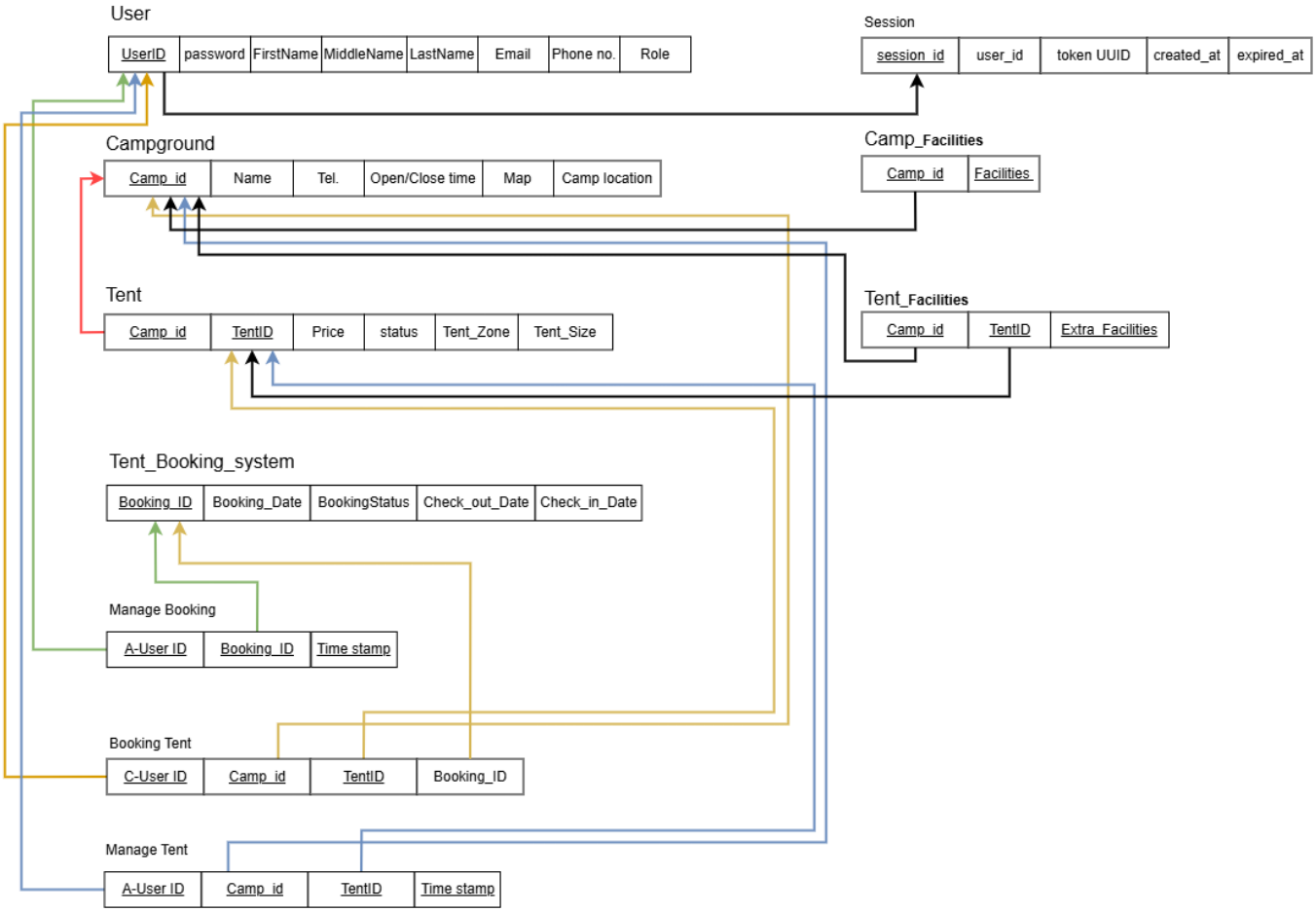
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

ER Diagram (Chen’s notation)



Schema diagram



SQL commands

Table

```
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    middle_name VARCHAR(100),
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    role VARCHAR(10) NOT NULL CHECK (role IN ('admin', 'customer'))
);

CREATE TABLE Campgrounds (
    campground_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    map_url TEXT,
    open_time TIME NOT NULL,
    close_time TIME NOT NULL
);

CREATE TABLE Tents (
    campground_id INT,
    tent_id INT NOT NULL,
    tent_size VARCHAR(50) NOT NULL,
    tent_zone VARCHAR(50) NOT NULL,
    status VARCHAR(10) NOT NULL CHECK (status IN ('available',
'occupied')),
    price DECIMAL(10,2) NOT NULL,

        -- Composite Primary Key (campground_id + tent_id)
    PRIMARY KEY (campground_id, tent_id),

    -- Foreign Key Constraint
    FOREIGN KEY (campground_id) REFERENCES Campgrounds(campground_id) ON
DELETE CASCADE
);


CREATE TABLE Tent_Booking_System (
    booking_id SERIAL PRIMARY KEY,
    booking_status VARCHAR(10) NOT NULL CHECK (booking_status IN
```

```
('confirmed', 'cancelled')),
    booking_date DATE,
    check_in_date DATE,
    check_out_date DATE

);

CREATE TABLE Manage_Booking (
    booking_id INT,
    A_user_id INT,
    Time_stamp TIME,
    PRIMARY KEY (booking_id, A_user_id,Time_stamp),
    FOREIGN KEY (booking_id) REFERENCES Tent_Booking_System(booking_id)
ON DELETE CASCADE,
    FOREIGN KEY (A_user_id) REFERENCES users(user_id) ON DELETE CASCADE
);

CREATE TABLE Tent_Bookings (
    C_user_id INT,
    tent_id INT,
    campground_id INT,
    booking_id INT,
    PRIMARY KEY (C_user_id, tent_id,campground_id),
    FOREIGN KEY (C_user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (tent_id,campground_id) REFERENCES
Tents(tent_id,campground_id) ON DELETE CASCADE,
    FOREIGN KEY (booking_id) REFERENCES tent_booking_system(booking_id)
ON DELETE CASCADE
);

CREATE TABLE Manage_Tent (
    tent_id INT,
    A_user_id INT,
    campground_id INT,
    Time_stamp TIME,
       PRIMARY KEY (campground_id, tent_id, A_user_id),  -- Composite
primary key to uniquely identify each management relation

    FOREIGN KEY (campground_id, tent_id)  -- Reference to weak entity
Tent
        REFERENCES Tents(campground_id, tent_id)
        ON DELETE CASCADE,

    FOREIGN KEY (A_user_id)  -- Reference to User
        REFERENCES Users(user_id)
        ON DELETE CASCADE
```

```
);

CREATE TABLE Campground_Facilities (
    campground_id INT,
    facilities  VARCHAR(100),
    PRIMARY KEY (campground_id, facilities),  -- Composite primary key
    FOREIGN KEY (campground_id) REFERENCES Campgrounds(campground_id) ON
DELETE CASCADE


);


CREATE TABLE Tent_Facilities (
    campground_id INT,
    tent_id INT,
    Facilities  VARCHAR(100),
      PRIMARY KEY (campground_id, tent_id, facilities),  -- Composite
primary key
    FOREIGN KEY (campground_id, tent_id)
        REFERENCES Tents(campground_id, tent_id)
        ON DELETE CASCADE
);


CREATE TABLE Sessions (
    session_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(user_id),
    token UUID UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expired_at TIMESTAMP
);
```

Test-table

```sql
-- 1user
INSERT INTO Users (first_name, middle_name, last_name, email, password,
phone, role) VALUES
('Aitsayaphan','Sigma','Limmuangnil','aitsayaphan@example.com','hashedpa
ssword1','0234567891','customer'),
('Aphiwich','Sigma','Sangpet','aphiwich@example.com','hashedpassword2','
0234567892','admin'),
('Atipat','Sigma','Buranavatanachoke','atipat@example.com','hashedpasswo
rd3','0234567893','customer'),
('Sirikarn','Sigma','Fugsrimuang','sirikarn@example.com','hashedpassword
4','0234567894','customer'),
('Penpitcha','Sigma','Piyawaranont','penpitcha@example.com','hashedpassw
ord5','0234567895','admin'),
('Pasit','Sigma','Bunsophon','pasit@example.com','hashedpassword6','0234
567896','customer');


-- 2Campground
INSERT INTO Campgrounds (name, location, phone, map_url, open_time,
close_time) VALUES
('Sunny Campground', 'Chiangmai', '0987654321', 'http://map1.com',
'08:00', '20:00'),
('Rainy Campground', 'Bangkok', '0987654322', 'http://map2.com',
'08:00', '21:00'),
('Winter Campground', 'Songkhla', '0987654323', 'http://map3.com',
'09:00', '23:00');


-- 3Tent
INSERT INTO Tents (campground_id, tent_id, tent_size, tent_zone, status,
price) VALUES
(4,1, 'SizeSmall', 'Zone1', 'available', 21.54),
(4,2, 'SizeSmall', 'Zone1', 'available', 67.59),
(4,3, 'SizeMedium', 'Zone1', 'occupied', 41.56),
(4,4, 'SizeLarge', 'Zone1', 'available', 13.92),
(4,5, 'SizeSmall', 'Zone2', 'available', 96.26),
(4,6, 'SizeSmall', 'Zone2', 'available', 45.35),
(4,7, 'SizeLarge', 'Zone2', 'occupied', 63.64),
(5,1, 'SizeSmall', 'Zone1', 'available', 67.59),
(5,2, 'SizeMedium', 'Zone1', 'occupied', 41.56),
(5,3, 'SizeLarge', 'Zone1', 'occupied', 66.22),
(5,4, 'SizeSmall', 'Zone2', 'available', 67.59),
(6,1, 'SizeSmall', 'Zone1', 'available', 20.36),
```

```
(6,2,'SizeSmall', 'Zone1', 'occupied', 50.21),
(6,3,'SizeLarge', 'Zone1', 'occupied', 63.64),
(6,4,'SizeMedium', 'Zone2', 'occupied', 41.56),
(6,5,'SizeLarge', 'Zone2', 'occupied', 66.22);


-- 4Tent Booking System
INSERT INTO Tent_Booking_System (booking_id, booking_status,
booking_date, check_in_date, check_out_date) VALUES
(1, 'confirmed', '2025-01-01', '2025-02-01', '2025-02-05'),
(2, 'cancelled', '2025-01-10', '2025-02-03', '2025-02-06'),
(3, 'confirmed', '2025-01-11', '2025-02-05', '2025-02-07'),
(4, 'confirmed', '2025-01-15', '2025-02-07', '2025-02-08');

-- 6Tent Bookings
INSERT INTO Tent_Bookings (C_user_id, tent_id, campground_id,
booking_id) VALUES
(7, 1, 4, 1),
(7, 2, 4, 2),
(7, 1, 5, 3),
(8, 1, 6, 4);

-- 8 Campground Facilities
INSERT INTO Campground_Facilities (campground_id, Facilities) VALUES
(4, 'Restroom'),
(4, 'Parking'),
(4, 'WiFi'),
(4, 'Fire Pit'),
(5, 'Restroom'),
(5, 'Shower'),
(5, 'Electric Outlet'),
(6, 'Parking'),
(6, 'WiFi'),
(6, 'Campfire Area');

-- 9Tent Facilities
INSERT INTO Tent_Facilities (campground_id, tent_id, Facilities) VALUES
(4, 1, 'Fan'),
(4, 1, 'Mattress'),
(4, 2, 'Blanket'),
(4, 2, 'Fan'),
(4, 3, 'Light'),
(4, 3, 'Fan'),
(4, 4, 'Fan'),
(5, 1, 'Sleeping Bag'),
(5, 1, 'Camping Table'),
```

```
(5, 2, 'Fan'),
(5, 2, 'Heater'),
(5, 3, 'Light'),
(6, 1, 'Light'),
(6, 1, 'Fan'),
(6, 2, 'Blanket'),
(6, 2, 'Camping Table'),
(6, 2, 'Light');
```

SQL commands (ต่อ)

Functional requirements

*** โดยกลุ่มของเรา มีการเรียกรวม admin และ customer ว่า “user” แล้วแบ่งด้วย role และการจอง จะให้ข้อมูลที่ customer จอง เข้าระบบจองเต็นท์ (booking_tent) ***

1. The system shall allow a user to register by specifying the name, telephone number, email, and password.

```
CREATE OR REPLACE FUNCTION register(
    p_first_name VARCHAR(100)  ,
    p_middle_name VARCHAR(100),
    p_last_name VARCHAR(100)  ,
    p_email VARCHAR(100)   ,
    p_password VARCHAR(255)  ,
    p_phone VARCHAR(15)
)
RETURNS TEXT AS --แก้ void เป็น text ได้
$$
declare
id_count integer := 0;
BEGIN
    IF EXISTS (SELECT 1 FROM Users WHERE email = p_email) THEN
        RAISE EXCEPTION 'Email is already registered';
    END IF;

    INSERT INTO Users (first_name, middle_name, last_name, email,
password, phone, role)
    VALUES (p_first_name, p_middle_name, p_last_name, p_email,
p_password, p_phone, 'customer');
    RETURN 'Registered successfully';
END;
$$
LANGUAGE plpgsql;
```

2. After registration, the user becomes a registered user, and the system shall allow the user to log in to use the system by specifying the email and password.

```
CREATE OR REPLACE FUNCTION login_user(
    p_email VARCHAR(100),
    p_password VARCHAR(255)
)
RETURNS TEXT AS
$$
DECLARE
    v_user_id INT;
    v_stored_password VARCHAR(255);
    v_token UUID;
BEGIN
    -- ตรวจสอบว่ามีอีเมลใน Users Table
    SELECT password INTO v_stored_password
    FROM Users
    WHERE email = p_email;

    IF NOT FOUND THEN
        RETURN 'Invalid email address';
    END IF;
    IF v_stored_password != p_password THEN
      RETURN 'Invalid password';
    END IF;

    -- create token for session --
    CREATE EXTENSION IF NOT EXISTS "pgcrypto";

     v_token := gen_random_uuid();  -- ต้องเปิดใช้ extension pgcrypto
หรือใช้ uuid_generate_v4()

    -- บันทึก Token ลงในตาราง Sessions
    INSERT INTO Sessions (user_id, token)
    VALUES (v_user_id, v_token);
    -- ส่งคืน Token ให้ผู้ใช้
    RETURN v_token::TEXT;
    -- **** ส่วนนี้ยังไม่ทำการแยก customer กับ admin
    RETURN 'Login successful';
END;
$$
LANGUAGE plpgsql;
```

The system shall allow a registered user to log out.

```
CREATE OR REPLACE FUNCTION logout_user(
    p_token UUID
)
RETURNS TEXT AS
$$
DECLARE
    v_session_id INT;
BEGIN
    -- ตรวจสอบว่า token นี้มีอยู่ในตาราง Sessions หรือไม่
    SELECT session_id INTO v_session_id
    FROM Sessions
    WHERE token = p_token;

    -- ถ้าไม่เจอ token
    IF NOT FOUND THEN
        RETURN 'Invalid session or already logged out';
    END IF;

    -- ลบ session ออกจากตาราง
    DELETE FROM Sessions
    WHERE session_id = v_session_id;

    RETURN 'Logout successful';
END;
$$
LANGUAGE plpgsql;
```

3. After login, the system shall allow the registered user to book up to 3 nights by specifying the date and the preferred campground. The campground list is also provided to the user. A campground information includes the campground name, address, and telephone number.

```
CREATE OR REPLACE FUNCTION booking_tent(
    p_user_id INT,
    p_tent_id INT,
    p_campground_id INT,
    p_check_in DATE,
    p_check_out DATE
)
RETURNS TEXT AS
$$
DECLARE
    v_booking_id INT;
    v_status VARCHAR(20);
    v_nights INT;
BEGIN
    -- ตรวจสอบจำนวนคืนที่จอง (ต้องไม่เกิน 3 คืน)
    SELECT (p_check_out - p_check_in) INTO v_nights;
    IF v_nights > 3 THEN
        RETURN 'You can only book a tent for up to 3 nights.';
    END IF;

    -- ตรวจสอบสถานะของเต็นท์
    SELECT status INTO v_status
    FROM tents
    WHERE tent_id = p_tent_id;

    -- ถ้าสถานะเป็น 'occupied' จะไม่สามารถจองได้
    IF v_status = 'occupied' THEN
        RETURN 'This tent is already occupied. Please choose
another tent.';
    END IF;

    INSERT INTO tent_booking_system (booking_status, booking_date,
check_in_date, check_out_date)
    VALUES ('confirmed', CURRENT_DATE, p_check_in, p_check_out)
    RETURNING booking_id INTO v_booking_id;
```

```
    INSERT INTO tent_bookings (C_user_id, tent_id, campground_id,
booking_id)
    VALUES (p_user_id, p_tent_id, p_campground_id, v_booking_id);

    UPDATE tents
    SET status = 'occupied'
    WHERE tent_id = p_tent_id;

    RETURN format('Tent booking successful! Your booking ID is
%s', v_booking_id);
END;
$$
LANGUAGE plpgsql;

-- เพิ่มข้อมูลลงเต้นท์ก่อน
INSERT INTO Tents (campground_id, tent_size, tent_zone, status,
price) --extra_facilities,
VALUES
(1, '2-person', 'A', 'available', 500), --'Near lake',
(1, '4-person', 'B', 'available', 800), --'Mountain view',
(1, '6-person', 'C', 'available', 1200); --'Near waterfall',

-- ทำการจอง (ต้องแน่ใจว่ามี user_id, tent_id, campground_id นั้นจริงๆก่อน แล้วค่อยจอง)
-- SELECT booking_tent(user_id, tent_id, campground_id,
check_in_date, check_out_date);
SELECT booking_tent(6, 5, 1, '2025-02-10', '2025-02-12');

--เรียกดูทั้งหมด (ตัวอย่าง)
SELECT
    tbs.booking_id,
    tbs.booking_status,
    tbs.booking_date,
    tbs.check_in_date,
    tbs.check_out_date,
    tb.user_id,
    tb.tent_id,
    tb.campground_id
FROM tent_booking_system tbs
JOIN tent_bookings tb ON tbs.booking_id = tb.booking_id
WHERE tb.user_id = 6;
```

4. The system shall allow the customer to view his campground bookings.

```sql
CREATE OR REPLACE FUNCTION user_view_bookings(p_user_id INT)
RETURNS TABLE (
    booking_id INT,
    booking_status VARCHAR(20),
    booking_date DATE,
    check_in_date DATE,
    check_out_date DATE,
    user_id INT,
    tent_id INT,
    campground_id INT,
    campground_name VARCHAR(255),
    campground_location VARCHAR(255)
) AS
$$
BEGIN
    RETURN QUERY
    SELECT
        tbs.booking_id,
        tbs.booking_status,
        tbs.booking_date,
        tbs.check_in_date,
        tbs.check_out_date,
        tb.c_user_id,
        tb.tent_id,
        tb.campground_id,
        c.name AS campground_name,    -- ดึงชื่อแคมป์
        c.location AS campground_location  -- ดึงที่ตั้งแคมป์
    FROM tent_booking_system tbs
    JOIN tent_bookings tb ON tbs.booking_id = tb.booking_id
    JOIN campgrounds c ON tb.campground_id = c.campground_id
-- JOIN กับตาราง campground
    WHERE tb.c_user_id = p_user_id;
END;
$$
LANGUAGE plpgsql;
```

5. The system shall allow the customer to edit his campground bookings.

```
CREATE OR REPLACE FUNCTION edit_booking_dates(
            -- ID ของการจองที่ต้องการแก้ไข
    p_user_id INT,
    p_booking_id INT,            -- ผู้ใช้ที่เป็นเจ้าของการจอง
    p_new_check_in DATE,      -- วันเช็คอินใหม่
    p_new_check_out DATE      -- วันเช็คเอาท์ใหม่
)
RETURNS TEXT AS $$
DECLARE
    v_nights INT;             -- ตัวแปรเก็บจำนวนคืนที่จอง
    v_existing_booking INT;   -- ตรวจสอบว่ามีการจองของ user นี้อยู่หรือไม่
BEGIN
    -- ตรวจสอบจำนวนคืนที่จอง (ต้องไม่เกิน 3 คืน)
    SELECT (p_new_check_out - p_new_check_in) INTO v_nights;
    IF v_nights > 3 THEN
        RETURN 'Error: You can only book a tent for up to 3
nights.';
    END IF;

    -- ตรวจสอบว่า booking_id นี้เป็นของ user จริงหรือไม่
    SELECT COUNT(*) INTO v_existing_booking
    FROM Tent_Booking_System tbs
    JOIN Tent_Bookings tb ON tbs.booking_id = tb.booking_id
    WHERE tb.booking_id = p_booking_id AND tb.C_user_id =
p_user_id;

    IF v_existing_booking = 0 THEN
        RETURN 'Error: Booking not found or does not belong to
this user.';
    END IF;

    -- อัปเดตวันที่เช็คอินและเช็คเอาท์
    UPDATE Tent_Booking_System
    SET check_in_date = p_new_check_in, check_out_date =
p_new_check_out
    WHERE booking_id = p_booking_id;
    RETURN 'Booking dates updated successfully!';
END;
$$ LANGUAGE plpgsql;
```

6. The system shall allow the customer to delete his campground bookings.

```
CREATE OR REPLACE FUNCTION cancel_tent_booking(
    p_booking_id INT
)
RETURNS TEXT AS
$$
DECLARE
    v_tent_id INT;
BEGIN
    -- ตรวจสอบว่า booking_id ที่ระบุมีการจองอยู่ในระบบหรือไม่
    IF NOT EXISTS (SELECT 1 FROM tent_booking_system WHERE
booking_id = p_booking_id) THEN
        RETURN 'Booking ID not found.';
    END IF;

    -- ดึง tent_id ที่ถูกจองมาจากการจอง
    SELECT tent_id INTO v_tent_id
    FROM tent_bookings
    WHERE booking_id = p_booking_id;

    -- ลบข้อมูลการจองจากทั้ง 2 ตาราง tent_bookings และ tent_booking_system
    DELETE FROM tent_bookings WHERE booking_id = p_booking_id;
    DELETE FROM tent_booking_system WHERE booking_id =
p_booking_id;

    -- อัพเดตสถานะของเต็นท์กลับเป็น 'available'
    UPDATE tents
    SET status = 'available'
    WHERE tent_id = v_tent_id;

    -- แจ้งผลการยกเลิกการจอง
    RETURN 'Tent booking canceled and status updated to
available.';
END;
$$
LANGUAGE plpgsql;
```

7. The system shall allow the admin to view any campground bookings.

```
CREATE OR REPLACE FUNCTION admin_view_bookings()
RETURNS TABLE(
    booking_id INT,
    user_id INT,
    first_name VARCHAR(100),
     middle_name VARCHAR(100),
     last_name VARCHAR(100),
     campground_id INT,
    campground_name VARCHAR(100),
     booking_date DATE,
    check_in_date DATE,
    check_out_date DATE,
     tent_id INT,
    booking_status VARCHAR(20)
) AS
$$
BEGIN
    RETURN QUERY
    SELECT
     tbs.booking_id,
          u.user_id,
          u.first_name,
          u.middle_name,
          u.last_name,
     tb.campground_id,
          c.name,
          tbs.booking_date,
          tbs.check_in_date,
     tbs.check_out_date,
          tb.tent_id,
          tbs.booking_status
     FROM tent_booking_system tbs
     JOIN tent_bookings tb ON tbs.booking_id = tb.booking_id
     JOIN campgrounds c ON tb.campground_id = c.campground_id
     JOIN users u ON tb.c_user_id = u.user_id;
END;
$$
LANGUAGE plpgsql;
```

8. The system shall allow the admin to edit any campground bookings.

```
CREATE OR REPLACE FUNCTION EditCampgroundBooking (
    admin_id INT,
    c_booking_id INT,
    c_booking_status VARCHAR(10)
)
RETURNS TEXT AS
$$
DECLARE
    user_what_role int ;
BEGIN
    IF c_booking_status NOT IN ('confirmed', 'cancelled') THEN
        RAISE EXCEPTION 'Invalid booking status. Use "confirmed"
or "cancelled".';
    END IF;

    SELECT COUNT(*) into user_what_role FROM Users WHERE user_id =
admin_id and role='admin';
    IF  ( user_what_role !=0 ) THEN
        -- Update booking status
        UPDATE tent_booking_system
        SET booking_status = c_booking_status
        WHERE booking_id = c_booking_id;

        INSERT INTO Manage_Booking (booking_id, A_user_id,
Time_stamp)
        VALUES (c_booking_id,admin_id, NOW());

        IF NOT FOUND THEN
            RAISE EXCEPTION 'Booking ID % not found.',
c_booking_id;
        END IF;
    ELSE

        RAISE EXCEPTION 'Access Denied: User % is not an admin.',
admin_id;
    END IF;
RETURN c_booking_status || ' booking successful!';
END;
$$ LANGUAGE plpgsql;
```

9. The system shall allow the admin to delete any campground bookings.

```
CREATE OR REPLACE FUNCTION DeleteCampgroundBooking (
    admin_id INT,
    c_booking_id INT
)
RETURNS TEXT AS
$$
BEGIN
    -- Check if the user is an admin
    IF EXISTS (SELECT 1 FROM Users WHERE user_id = admin_id AND
role = 'admin') THEN

        INSERT INTO Manage_Booking (booking_id, A_user_id,
Time_stamp)
        VALUES (c_booking_id,admin_id, NOW());

        -- Delete from tent_bookings first if there's a foreign
key dependency
        DELETE FROM tent_bookings
        WHERE booking_id = c_booking_id;

        -- Then delete from tent_booking_system
        DELETE FROM tent_booking_system
        WHERE booking_id = c_booking_id;



    ELSE
        -- Raise an error if the user is not an admin
        RAISE EXCEPTION 'Access Denied';
    END IF;
RETURN 'Delete booking of customer number '|| c_booking_id || '
SUCCESSFULL';
END;
$$ LANGUAGE plpgsql;

--DeleteCampgroundBooking(admin_id ,c_booking_id )
-- select DeleteCampgroundBooking(1 ,6 ) --not admin
-- select DeleteCampgroundBooking(5 ,6 ) --admin
CREATE OR REPLACE FUNCTION EditCampgroundBooking (
```

```
    admin_id INT,
    c_booking_id INT,
    c_booking_status VARCHAR(10)
)
RETURNS TEXT AS
$$
DECLARE
    user_what_role int ;
BEGIN
    -- Check if the booking status is valid
    IF c_booking_status NOT IN ('confirmed', 'cancelled') THEN
        RAISE EXCEPTION 'Invalid booking status. Use "confirmed"
or "cancelled".';
    END IF;

    -- Check if the user is an admin
    SELECT COUNT(*) into user_what_role FROM Users WHERE user_id =
admin_id and role='admin';
    IF  ( user_what_role !=0 ) THEN

        -- Update the booking status
        UPDATE tent_booking_system
        SET booking_status = c_booking_status
        WHERE booking_id = c_booking_id;
        INSERT INTO Manage_Booking (booking_id, A_user_id,
Time_stamp)
        VALUES (c_booking_id,admin_id, NOW());
        -- Optionally, check if the update was successful
        IF NOT FOUND THEN
            RAISE EXCEPTION 'Booking ID % not found.',
c_booking_id;
        END IF;

    ELSE
        -- Raise an error if the user is not an admin
        RAISE EXCEPTION 'Access Denied: User % is not an admin.',
admin_id;
    END IF;
RETURN c_booking_status || ' booking successful!';
END;
$$ LANGUAGE plpgsql;
```

SQL complex query

Most popular campground - top 3 Query แสดงข้อมูลรายการ campground ที่มี user จองมากที่สุด 3 อันดับแรก โดยมีรายละเอียดข้อมูลดังนี้

1. Campground name
2. Campground telephone
3. Campground location
4. Open - time
5. Close – time
6. Campground facilities
7. จำนวนการจองของ campground

```
SELECT
    C.name AS Campground,
    C.location AS Location,
    C.map_url AS Map,
     COALESCE(F.facilities, 'No Facilities') AS Facilities,
    C.open_time AS Open_Time,
    C.close_time AS Close_Time,
    COALESCE(B.booking_made, 0) AS Booking_made
FROM campgrounds C
LEFT JOIN (
    SELECT
        campground_id,
        COUNT(*) AS booking_made
    FROM tent_bookings
    GROUP BY campground_id
) B ON C.campground_id = B.campground_id
LEFT JOIN (
    SELECT
        campground_id,
        STRING_AGG(DISTINCT facilities, ', ') AS facilities
    FROM campground_facilities
    GROUP BY campground_id
) F ON C.campground_id = F.campground_id
ORDER BY booking_made DESC
LIMIT 3;
```

Document-based design schema

```
{
    "title": "campground",
    "required": ["name", "location", "phone", "open_time",
"close_time", "tents"],
    "properties": {
        "_id": { "bsonType": "objectId" },
        "name": { "bsonType": "string" },
        "location": { "bsonType": "string" },
        "phone": {
            "bsonType": "string",
            "pattern": "/^[0-9]{3}-[0-9]{3}-[0-9]{4}$/"
        },
        "map_url": { "bsonType": "string" },
        "open_time": { "bsonType": "string" },
        "close_time": { "bsonType": "string" },
        "tents": {
            "bsonType": "array",
            "items": {
                "bsonType": "object",
                "properties": {
                    "tent_id": { "bsonType": "string" },
                    "tent_size": { "bsonType": "string" },
                    "tent_zone": { "bsonType": "string" },
                    "status": {
                        "bsonType": "string",
                        "enum": ["available", "occupied"]
                    },
                    "price": { "bsonType": "decimal" }
                }
            }
        }
    }
}
```